

# NT Object (In)Security

A look into one of NT's most undocumented subsystems

Riley Hassell

SOURCE Boston 2010



<https://www.isecpartners.com>

# Why?

- Escalation in the NT environment is much easier given real world software deployments
- “Crapware” makes for easily escalation
- Funnel effect

# Overview

- The Object Manager
- Object Directory Security Controls
- Object Security Controls
- Object Squatting
- ObjectTrace

# The Object Manager

# What is the Object Manager

- It manages all the NT objects used by the Windows Operating System
- Supports unnamed and named objects using a directory data structure that in many ways resembles a file system (more on this later )
- It associates objects with processes using per process handle tables
- The object manager is one of the most undocumented subsystems in Microsoft Windows

# What is an NT Object?

- Device
  - Directory
  - Event
  - File
  - Key
  - LPC/ALPC Ports
  - Mutant
  - Section
- ...and many more

# Object Manager Kernel Routines

NT Kernel routines have Obxxx prefix:

- ObCreateObject
- ObCreateObjectType
- ObCheckObjectAccess
- ObInsertObject
- ObQueryNameString
- ObGetObjectSecurity
- ObOpenObjectByName
- ...

Few of these API(s) are exported so working with the Object Manager Directory is difficult at best

# Object Directory



# Named Objects

- Most user API(s) offer a name parameter to be used for the new name of the specific object being created
- Developers can name objects so they can be accessed by other processes thus creating an IPC mechanism
- Named objects are organized in the object manager directory

# Object Directory

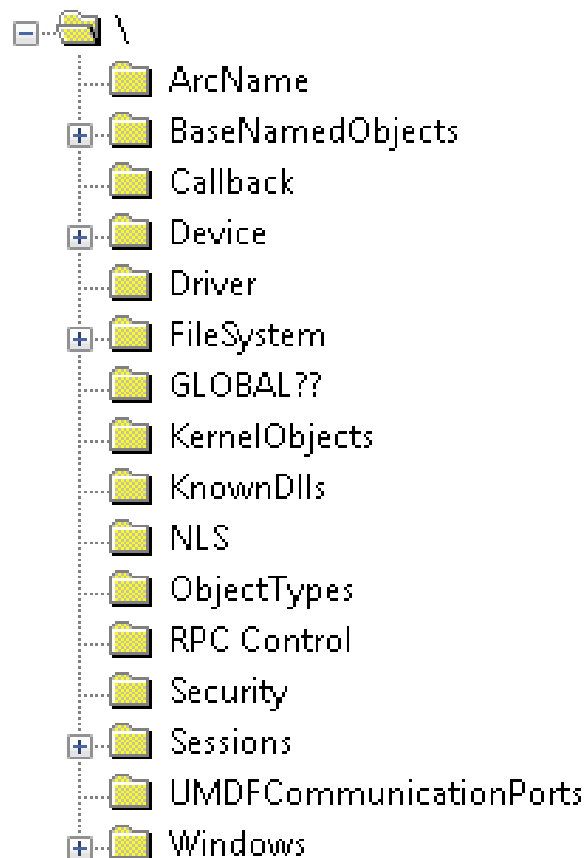


Figure 1. Snapshot take from WinObj navigation pane

# Choosing a name...

“The name can have a "Global\" or "Local\" prefix to explicitly create the object in the global or session name space. **The remainder of the name can contain any character except the backslash character (\).**”

-- CreateEvent Function @ MSDN Library

# Same feeling when I saw this...

...The answers to your personal Security Questions are case sensitive and cannot contain any special characters like an apostrophe, or the words "insert," "delete," "drop," "update," or "select."

--Anonymous Banking Site F.A.Q

# Local Objects

Local objects reside in:

`"\Sessions\n\BaseNamedObjects"`  
(where  $n$  is current session id)

No prefix needed:

E.g. `CreateEvent(x,x,x,"ISECEVENT");`

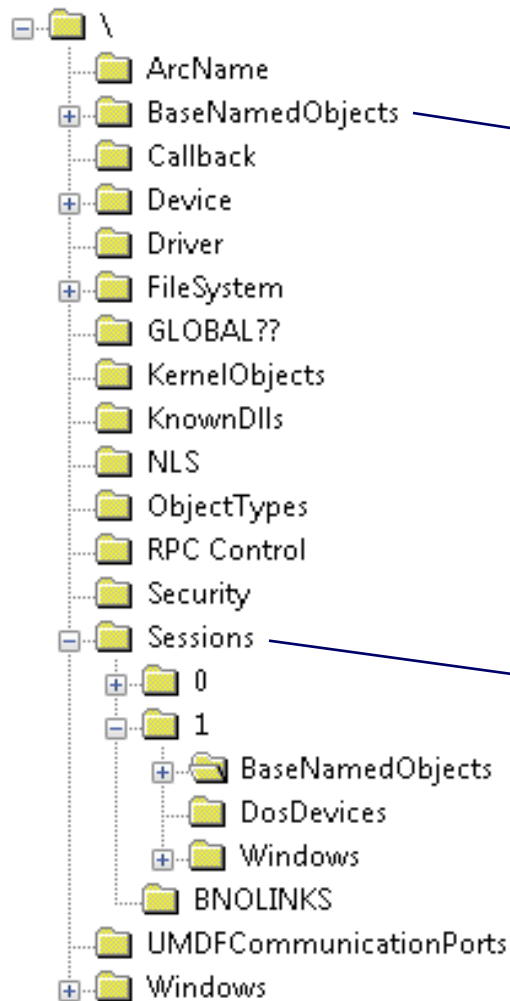
# Global Objects

Global objects reside in “\BaseNamedObjects”

Accessed via Global prefix

E.g. `CreateEvent(x,x,x, "Global\ISECEVENT");`

# Object Manager Directory



**“\BaseNamedObjects”**

**Global objects are created here. User prefixes  
“Global\” to object name during creation.**

**E.g. “\BaseNamedObjects\ISECTEST”**

**“\Sessions”**

**Session instances. Local objects reside here.**

**E.g. “\Sessions\1\BaseNamedObjects\ISECTEST”**

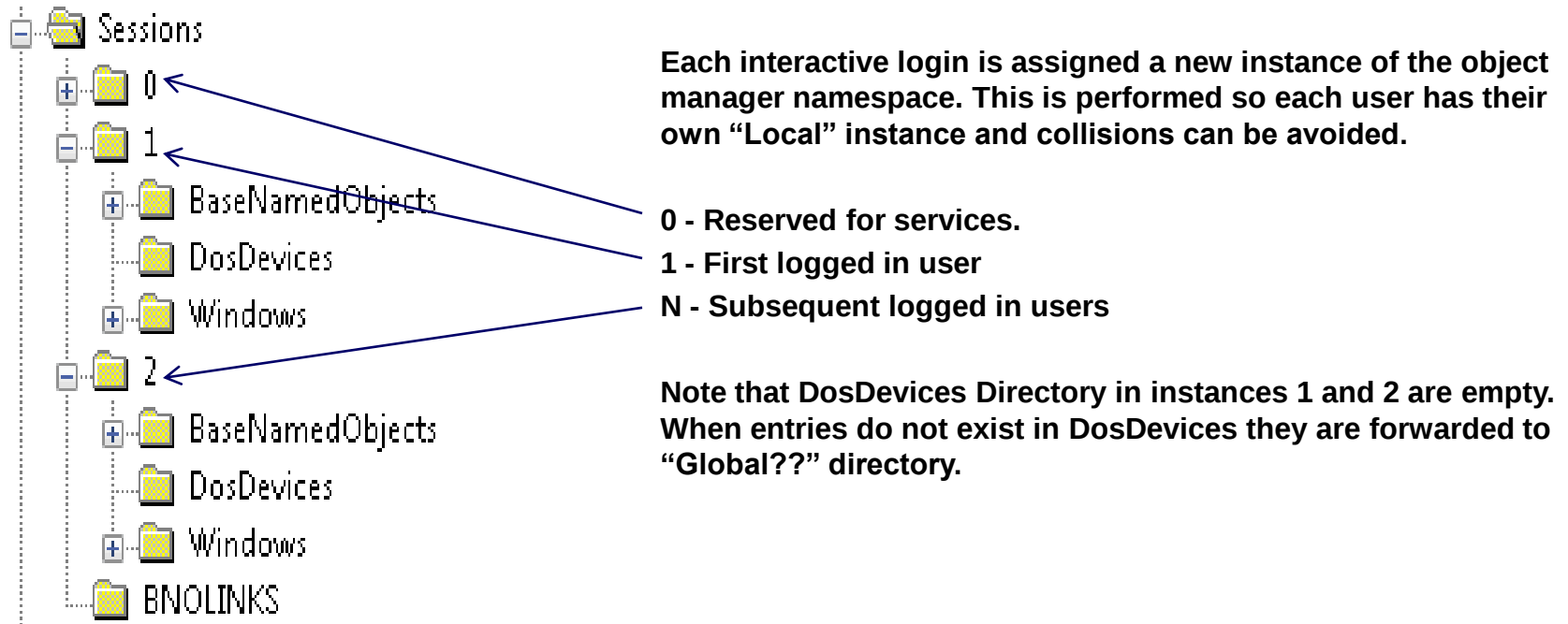
# Session Isolation

“Session Directories are isolated from each other, and administrative privileges are required to create a global object (except for section objects).”

--Windows Internals 5<sup>th</sup> Edition



# Name space instances



# Object Directory Security Controls

# Directory Permissions

```
C:\>accesschk -ruo everyone \
```

```
Accesschk v4.23 - Reports effective permissions for securable objects  
Copyright (C) 2006-2008 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

```
R [Directory] \ArcName  
R [Directory] \Device  
R [Directory] \Windows  
R [Directory] \Sessions  
RW [Directory] \RPC Control  
R [Directory] \KernelObjects  
RW [Directory] \BaseNamedObjects  
R [Directory] \GLOBAL??  
R [Directory] \Callback  
R [Directory] \KnownDlls
```

# Bypass Traverse Checking

Bypass Traverse Checking determines which users can traverse an object hierarchy even though they might not have permissions on the parent levels of the traversed hierarchy.

# Bypass Traverse Checking - Cont

“At this point you might believe that Bypass Traverse Checking is a security hole, but most of the time security settings on a directory are inherited by files and subdirectories within it. Thus, if a user doesn’t have access to a directory they won’t have access to items within it, even if they happen to know the names of those items, unless someone explicitly grants them access.”

-- *Mark Russinovich*

# Bypass Traverse Checking - Cont

- Object creators often add explicit permissions to enable IPC
- While the Object Manager Directory shares many similarities of NT file systems it does not come with all the bells and whistles... like the inheritance of permissions ;)

# Namespace Traversal

Local session objects are accessible by other users

## DEMO

# What does this mean?

- Any application that creates a local object with poor security DACL configuration opens an avenue of attack
- iSEC identified many issues in COTS where this is a problem
- Serious problem in OEM computers due to “Crapware”



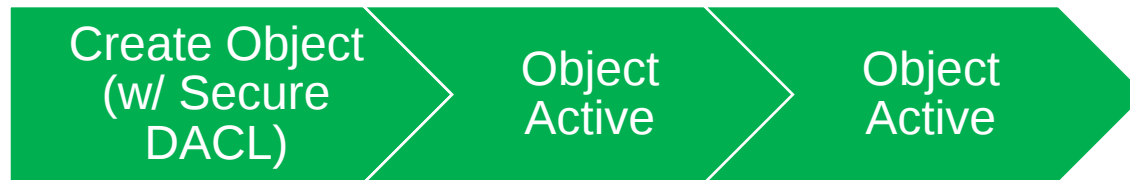
# Object Security Controls

# Configuring Object Security

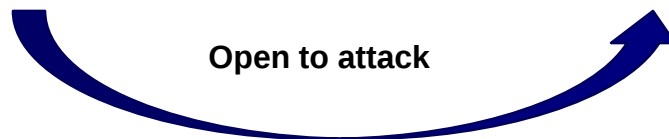
- SecurityDescriptor passed in object creation routine.
- Alter the default DACL so all objects are created with an intended DACL.
- Supply new DACL to existing object using set security routines:
  - SetKernelObjectSecurity()
  - SetPrivateObjectSecurity()
  - SetUserObjectSecurity()

# Create and Apply Dangers

- Create With Secure DACL



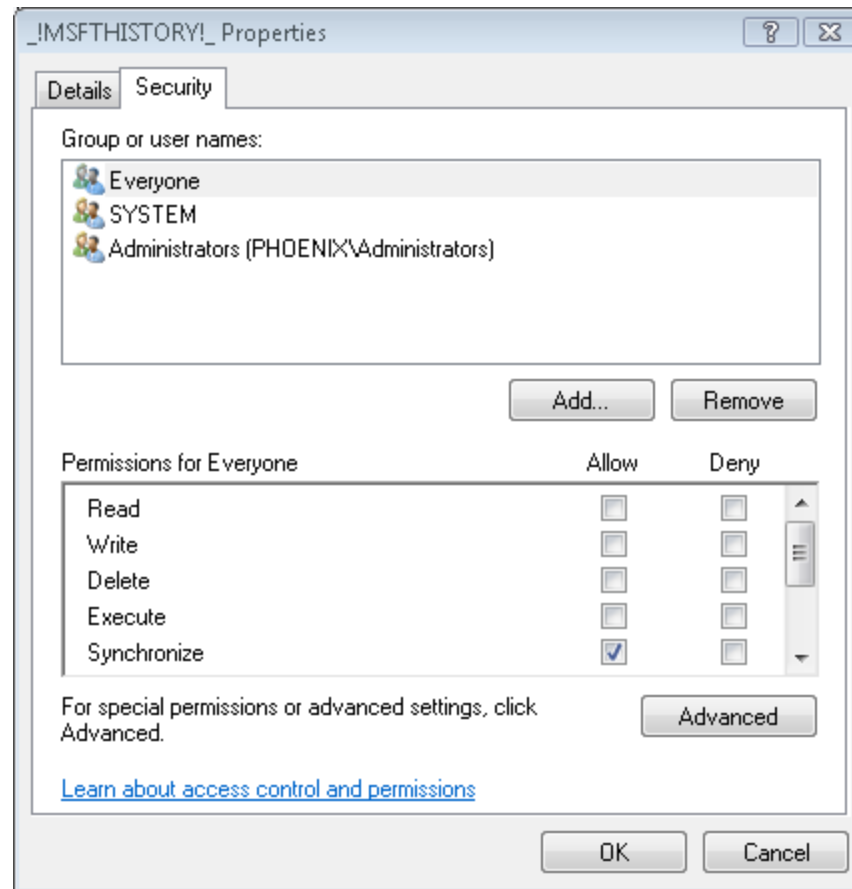
- Create With Secure DACL



# Object Security

- Object security is governed through security descriptors
- Security Descriptors contain information about the owner and group of the object and who can access the object (DACL)
- Most object API(s) allow a developer to supply a security descriptor for an object during creation
- If a DACL is not supplied the default DACL from primary token of the process creator

# Object Security - EditSecurity



# DACL

The discretionary access control list (DACL) on a kernel object determines which users can access the object. This list contains a number of access control entries (ACEs) which either grant or deny a user or group access to the object.

# Default DACL

- Rather than explicitly constructing a DACL for an object, it is a common practice to create the object by using the default DACL for the process. Typically, this will allow only the creator and the system to obtain access to the object. To grant other users access to a kernel object (mutex, event, file mapping, and so forth) that is created under your security context, you can do either of the following:
  - Explicitly add an access-allowed ACE to the object's DACL.
  - Modify the default DACL for the process before creating the object.

# Default DACL Dangers

- Process based therefore the change will affect other object creations in the process
- What if an out of process RPC Service or application plugin changed the default DACL to allow access to everyone?



# Finding Unprotected Named Objects

**AccessChk v4.23**

<http://technet.microsoft.com/en-us/sysinternals/bb664922.aspx>

To see all global objects that Everyone can modify:

**accesschk -wuo everyone \BaseNamedObjects\**

To see all local objects that Everyone can read in the first session:

**accesschk -ruo everyone \Sessions\1\BaseNamedObjects\**

# Unnamed Objects

# Unnamed Objects

- Certain unnamed objects are forcefully named by their associated subsystem
- These objects are accessible!?! ;)

# RPC Name Generation

- RPC (LRPC-1e0245833c186937f8)

LRPC\_ADDRESS::ServerSetupAddress

```
...  
.text:778454E5          push     9  
.text:778454E7          lea      eax, [ebp+var_10]  
.text:778454EA          push     eax  
.text:778454EB          mov     [ebp+var_48], ecx  
.text:778454EE          mov     [ebp+var_44], dx  
.text:778454F2          call    GenerateRandomNumber(uchar *,int)  
.text:778454F7          test    eax, eax  
.text:778454F9          jnz     loc_77887EA6  
.text:778454FF          lea     ecx, [ebp+var_42]  
...
```

- Same functionality exists for OLE Endpoints

# Accessing “Unnamed” Endpoints

```
C:\Users\Riley>ifids -e LRPC-1e0245833c186937f8 PHOENIX -p ncalrpc
```

```
Interfaces: 31
```

```
00000134-0000-0000-c000-000000000046 v0.0
```

```
18f70770-8e64-11cf-9af1-0020af6e72f4 v0.0
```

```
00000131-0000-0000-c000-000000000046 v0.0
```

```
00000143-0000-0000-c000-000000000046 v0.0
```

```
00000132-0000-0000-c000-000000000046 v0.0
```

```
...
```

# Not Enforced on Existing Handles

Routines:

- SetKernelObjectSecurity
- SetPrivateObjectSecurity
- SetUserObjectSecurity

# Object Squatting

# SeCreateGlobalPrivilege

Starting with Windows Server 2003, Windows XP SP2 and Windows 2000 Server SP4, the creation of a file-mapping object (using [CreateFileMapping](#)) from a session other than session zero is a privileged operation.

The privilege check is limited to the creation of file mapping objects and does not apply to opening existing objects, i.e. a user without the SeCreateGlobalPrivilege can still open an existing section object.



# SeCreateSymbolicLinkPrivilege

- Required to create a symbolic link
- Only Administrators typically have this privilege

# Those other objects...

Limited users can still create global objects of types other than those that are restricted as the unrestricted object types are deemed less dangerous. Ie. LPC Ports, Pipes, Events, etc...

# Object Squatting

- Non-administrators can create objects in global “\BaseNamedObiects” and in “\RPC Controls”
- Rogue user can create an object that masquerades as legitimate object by creating an object with the same name
- Disrupt user services and potentially escalate privilege

# Object Squatting

DEMO

# Object Squatting and Impersonation

On older systems (pre Windows 2003 and XP SP2) direct client impersonation is possible. A rogue server could impersonate a connecting client privilege level. Often acquiring LOCAL\_SYSTEM privilege. Modern systems require that the identity have the privilege *SeImpersonatePrivilege*.

*On a current Vista Enterprise with .NET 3.5 the following users have this privilege:*

- Administrators
- ASPNET
- LOCAL\_SERVICE
- NETWORK\_SERVICE
- SERVICE

# Attack and escalate

- LOCAL SERVICE, NETWORK SERVICE, and SERVICE all have SeCreateGlobalPrivilege and *SeImpersonatePrivilege*
- An attacker that can compromise a server process with reduced privilege, but with *SeImpersonatePrivilege* enabled, may be able to squat on a known endpoint and impersonate higher privileged callers

# Attack and Escalate - Cont

- If an attacker can get `SelImpersonatePrivilege` it's generally assumed they can escalate privilege to `SYSTEM` through a variety of means

# Mutex Objects

“If *lpName* matches the name of an existing named mutex object, this function requests the `MUTEX_ALL_ACCESS` access right. In this case, the *bInitialOwner* parameter is ignored because it has already been set by the creating process. If the *lpMutexAttributes* parameter is not `NULL`, it determines whether the handle can be inherited, but its security-descriptor member is ignored.”

MSDN - CreateMutex Function

[http://msdn.microsoft.com/en-us/library/ms682411\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms682411(VS.85).aspx)



# Protecting yourself

- Avoid altering object security after creation. Tighten DACL at time of creation
- Assess DACLs on product owned objects and review for weak DACL configuration
- Reduce attack surface area by removing unused and/or unneeded objects
- Tighten existing permissions. Typically a DACL will only need to provide access to LOCAL\_SYSTEM, Administrator and the object creator
- Consider using private namespaces:  
[http://msdn.microsoft.com/en-us/library/ms684295\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms684295(VS.85).aspx)

# Private Namespaces

- Introduced in Windows Vista and Windows Server 2008
- Developers can create a private namespace prefix only accessible by authorized members with knowledge of the namespace.
- Custom prefix is isolated using a boundary descriptor
- A boundary descriptor is populated with SID(s) of authorized parties that can access the name space.

# Tools

- Application Verifier  
Has included support for object squatting detection and checks since 2004:
  - <http://technet.microsoft.com/en-us/library/cc700837.aspx>
- WinObj  
Tool used to view the object directory
  - <http://technet.microsoft.com/en-us/sysinternals/bb896657.aspx>
- AccessChk  
This tool is great for enumerating open permissions in the object directory
  - <http://technet.microsoft.com/en-us/sysinternals/bb664922.aspx>
- ObjectTrace – iSEC Partners...?

# ObjectTrace

- Allows the tester to monitor creation of all system objects and locate objects susceptible to squatting or compromise
- Implemented via object manager callback support in Vista SP1, Windows Server 2008
- Object manager callbacks only intended for processes and threads
- An undocumented flag on default object types is modified dynamically to permit monitoring of all object types

# ObjectTrace

DEMO

# ObjectTrace 1.0

See iSEC Tools for more information:

iSEC Tools

<https://www.isecpartners.com/tools.html>

# Conclusion

- We often make assumptions about these subsystems that are far from the true
- Most applications create objects in a way that is not secure
- “Crapware” demolishes the local security of your system
- Object management systems can be used by attackers to escalate privilege and disrupt even heavily hardened systems

# Thanks

- Jesse Burnes (iSEC), Scott Stender (iSEC) and Brad Hill (iSEC) for advice and assistance with the murkier parts of NT access control
- Andreas Junestam (iSEC) and Geng Yang (iSEC) for extensive peer review and testing of ObjectTrace
- Steve Kogan for holding my hand through icon creation design. Scaled blending transparencies, seriously?
- To all my build engineer and interface designer friends that I've mocked over the years. Your job is much harder than I thought 😊



# Questions & Comments?

Riley Hassell - [riley@isecpartners.com](mailto:riley@isecpartners.com)